Software Development Cost Estimation Using Function Points

Jack E. Matson, Bruce E. Barrett, and Joseph M. Mellichamp

Abstract—**This paper presents an assessment of several published statistical regression models that relate software development effort to software size measured in function points. The principal concern with published models has to do with the number of observations upon which the models were based and inattention to the assumptions inherent in regression analysis. The research describes appropriate statistical procedures in the context of a case study based on function point data for 104 software development projects and discusses limitations of the resulting model in estimating development effort. The paper also focuses on a problem with the current method for measuring function points that constrains the effective use of function points in regression models and suggests a modification to the approach that should enhance the accuracy of prediction models based on function points in the future.**

Index Terms—**Function points, regression analysis, cost estimation**

I. Introduction

An increasingly important facet of software development is the ability to estimate the associated cost of development early in the development life cycle. The primary factor affecting software cost estimation is the size of the project; however, estimating software size is a difficult problem that requires specific knowledge of the system functions in terms of scope, complexity, and interactions [14]. A number of software size metrics are identified in the literature; the most frequently cited measures are lines of code and function point analysis.

A. Lines of Code

The traditional size metric for estimating software development effort and for measuring productivity has been lines of code (LOC). A large number of cost estimation models have been proposed, most of which are a function of lines of code, or thousands of lines of code (KLOC). Generally, the effort estimation model consists of two parts. One part provides a base estimate as a function of software size and is of the following form:

$$E = A + B \times (\text{KLOC})^C,$$

where E is the estimated effort in man-months; A, B, and C are constants; and KLOC is the estimated number of thousands of line of code in the final system. The second part modifies the base estimate to account for the influence of environmental factors [15]. Examples of environmental factors include the use of such practices as structured code, top-down design, structured walk-throughs, and chief programmer teams; personnel ability; and hardware constraints [5]. As an example, Boehm's [4] COCOMO model uses lines of code raised to a power between 1.05 and 1.20 to determine the base estimate. The specific exponent depends on whether the project is simple, average, or complex. The model then uses 15 cost influence factors as independent multipliers to adjust the base estimate. Conte, Dunsmore, and Shen [7] identified some typical models including the following:

$E = 5.2 \times (\text{KLOC})^{0.91}$	(Walston-Felix model)
$E = 5.5 + 0.73 \times (\text{KLOC})^{1.16}$	(Bailey-Basili model)
$E = 3.2 \times (\text{KLOC})^{1.05}$	(Boehm simple model)
$E = 3.0 \times (\text{KLOC})^{1.12}$	(Boehm average model)
$E = 2.8 \times (\text{KLOC})^{1.20}$	(Boehm complex model)
$E = 5.288 \times (\text{KLOC})^{1.047}$ for KLOC > 9	(Doty model).

The definition of KLOC is important when comparing these models. Some models include comment lines, and others do not. Similarly, the definition of what effort (E) is being estimated is equally important. Effort may represent only coding at one extreme or the total analysis, design, coding, and testing effort at the other extreme. As a result, it is difficult to compare these models.

There are a number of problems with using LOC as the unit of measure for software size. The primary problem is the lack of a universally accepted definition for exactly what a line of code really is. Jones [10] identified 11 major variations of line counting methods. Since few authors state the line-counting rules they used, much of the literature has an "uncertainty of perhaps 500% attributable to line counting variations." The variations make it very difficult to compare studies using lines of code as a measure of software size.

Another difficulty with lines of code as a measure of system size is its language dependence. It is not possible to directly compare projects developed by using different languages [16]. For example, the time per line for a high-level language may be greater than for a lower-level language. There is no way to accommodate the fact that fewer lines of code may be required for a higher-level language to provide the same function.

Still another problem with the lines of code measure is the fact that it is difficult to estimate the number of lines of code that will be needed to develop a system from the information

Manuscript received April 21, 1993; revised November 1993. Recommended by R. DeMillo.

J. E. Matson is with the Department of Industrial Engineering, University of Alabama, Tuscaloosa, AL 35487.

B. E. Barrett and J. M. Mellichamp are with the Department of Management Science and Statistics, University of Alabama, Tuscaloosa, AL 35487.

IEEE Log Number 9216525.

available at requirements or design phases of development [8]. If cost models based on size are to be useful, it is necessary to be able to predict the size of the final product as early and accurately as possible. Unfortunately, estimating software size using the lines of code metric depends so much on previous experience with similar projects that different experts can make radically different estimates [7]. Finally, the lines of code measure places undue emphasis on coding, which is only one part of the implementation phase of a software development project. Emrick [8] stated that coding accounts only for 10% to 15% of the total effort on a large development system and questioned whether the total effort is really linearly dependent on the amount of code.

B. Function Point Analysis

Function point analysis is a method of quantifying the size and complexity of a software system in terms of the functions that the system delivers to the user. The function delivered is unrelated to the language or tools used to develop a software project [2]. Function point analysis is designed to measure business-type applications; it is not appropriate for other types of applications such as technical or scientific applications. These applications generally deal with complex algorithms that the function point method is not designed to handle [24].

The function point approach has features that overcome the major problems with using lines of code as a measure of system size. First, function points are independent of the language, tools, or methodologies used for implementation; i.e., they do not take into consideration programming languages, data base management systems, processing hardware, or any other data processing technology [16], [24]. Second, function points can be estimated from requirements specifications or design specifications, thus making it possible to estimate development effort in the early phases of development [16]. Since function points are directly linked to the statement of requirements, any change of requirements can easily be followed by a reestimate [9]. Third, since function points are based on the system user's external view of the system, nontechnical users of the software system have a better understanding of what function points are measuring [12]. The method resolves many of the inconsistencies that arise when using lines of code as a software size measure.

Function points have been incorporated as an option in two commercially available software packages, SPQR/20 [11] and ESTIMACS^{TM} [12], [22]. SPQR/20 (software productivity, quality, reliability) is based on a modified function point method; ESTIMACS^{TM} contains a module which estimates function points. The primary difference in the SPQR/20 model and the traditional function point method is in the way complexity is handled. Whereas traditional function point analysis is based on evaluating 14 factors, SPQR/20 separates complexity into three categories: complexity of algorithms, complexity of code, and complexity of data structures. The SPQR/20 method makes it easier to evaluate the complexity factors (three questions as opposed to the detail of 14 factors). According to Porter [20], the SPQR/20 method did not seem to differ from function point analysis. The method is available in a commercial system, but documentation of the counting practices is not available in the public domain. Traditional function point analysis remains the industry standard, however, and is the method of choice of the International Function Point Users Group.

ESTIMACS^{TM} [22] is a proprietary system designed to give development effort estimates at the conception stage of a project. At this early phase, the full details of the system are not known, and normally only gross estimates are needed to make "go" and "no-go" decisions. In addition to estimated work effort, the system contains a module which will project the expected function points. This also is a very high-level estimate and generally is not very accurate [21].

In summary, function point analysis appears to have advantages over lines of code as a measure of software size for use in estimating software development cost, and there is widespread industry support for this method. Unfortunately, there are few published cost estimation models that use function points as the key input parameter.

Counting Function Points: Briefly, raw function counts are arrived at by considering a linear combination of five basic software components (inputs, outputs, master files, interfaces, and inquiries), each at one of three levels: low, average or high. We may express this as follows:

$$\text{Function Count} = \sum_{i=1}^{5}\sum_{j=1}^{3} w_{ij}z_{ij},$$

where z_{ij} is the count for component i at level j (e.g., outputs at high complexity) and w_{ij} is the fixed weight assigned by the Albrecht procedure. These function counts are also known as unadjusted function points (UFP). The final number of function points is arrived at by multiplying the UFP by an adjustment factor that is determined by considering 14 aspects of processing complexity. This adjustment factor allows the UFP count to be modified by at most ±35%. The final, adjusted, FP count for the k_{th} project is then the following:

$$\text{FP}_k = c_k \sum_{i=1}^{5}\sum_{j=1}^{3} w_{ij}z_{ij}, \qquad (1)$$

where c_k is between 0.65 and 1.35. (For a summary of the mechanics of function point counting, see [12]. For a more detailed account, see [23].)

Uses of Function Points: The collection of function point data has two primary motivations. One is the desire by managers to monitor levels of productivity, for example, number of function points achieved per work hour expended. From this perspective, the manager is not concerned with when the function point counts are made, but only that the function points accurately describe the "size" of the final software project. In this instance, function points have an advantage over LOC in that they provide a more objective measure of software size by which to assess productivity.

Another use of function points, which is the focus of this article, is in the *estimation* of software development cost. There are only a few previous studies that address this issue, though it is arguably the most important potential use of function point data. In Section II, we briefly review the goals and methodology of model selection and then examine two data sets and associated cost estimation models from the literature, pointing out some of the pitfalls of improper model selection that may arise primarily as a result of too few data points. In Section III we develop two cost estimation models utilizing a comparatively large data set ($n = 104$) and examine the predictive properties of each. Finally, in Section IV, we point out the limitations attendant with using function points to predict software size and propose a new direction for future efforts in the development of function point cost estimation models.

II. Estimation Models from the Literature

In developing a useful regression model, a number of concerns must be addressed. The first is model adequacy, or explanatory power of the independent variable(s) in accounting for the variability of the dependent variable. This is typically measured by the coefficient of multiple determination, R^2. However, a large value of R^2 is not the only measure of a good model. In some regard, it is not even the most important. Estimation theory for linear regression is tied to certain assumptions about the distribution of the residual or error terms. If these are seriously violated, a large R^2 may be of little importance.

These concerns come under the heading of model aptness, which refers to the conformity of the behavior of the residuals to the underlying assumptions about the errors in the model. Specifically, the usual assumptions for the error values in linear regression models are that these terms are distributed as independent, normal random variables with mean zero and identical variances. These assumptions are typically verified with the aid of diagnostic plots. Most common are the normal probability plot for verifying the normality of the residuals and a scatter plot of the residuals versus the fitted values to confirm the independence and homoscedasticity (*i.e.*, constant variance) of the residuals. When one or more of these assumptions is violated, transformation of variables is often attempted as a remedy.

A further concern is model stability, which refers to the resistance to change in the fitted model under small perturbations of the data. It is now generally recognized that residual analysis alone is inadequate in answering the questions of stability. This effect can be summarized by saying that the ordinary least-squares criterion gives disproportionately large weights to cases which are extreme in the predictor variables in determining the fit, often resulting in small residuals for those extreme or high leverage cases. Regression diagnostics, then, is generally understood to be the class of methods used to validate the probability assumptions about the errors, as well as the assessment of the stability of the fitted model, distinct from the probabilistic behavior of the errors.

By far, the most common approach for assessing model stability is case deletion. Each case (or data point) is removed in turn from the data and the various regression statistics, such as the estimated regression coefficients, the fitted values and the coefficient covariances, are recalculated. Cases whose removal substantially alter the results obtained using the full set of data are said to be influential. The most widely used influence measure is Cook's Distance [6], which measures changes in the estimated coefficients. This is the measure used in this study.

In assessing the influence of specific cases, we rely on a technique suggested by McCulloch and Meeter [18] and Barrett and Ling [3] to examine Cook's Distance. When investigating cases for their degree of influence, two components are of interest: a leverage component and a residual component. Since the influence may be expressed as a product of the leverage component and the residual component, cases which are large in one or both of these are candidates for high influence. For ease of display, the *logs* of the leverage and residual components are plotted. The contours of constant influence are straight lines with slope of -1, and the sum of the coordinates is the log of the influence.

A. The Albrecht and Gaffney Model

Albrecht and Gaffney [2] collected data on 24 applications developed by IBM Data Processing Services. Using a somewhat less refined counting method than that described in (1), they give the function point counts and the resulting work-hours, which we call effort, for each project. Ordinary least-squares regression was used to determine the fitted line for the dependent variable, E (effort), expressed as a function of the independent variable, FP. A scatter-plot of this data and the resulting regression function are shown in Fig. 1(a).

The explanatory power of the model is relatively high at $R^2 = 87.4\%$. However, the residuals are troublesome in several respects. From Fig. 1(b) we note serious autocorrelation; the first eight residuals are positive while eleven of the next fifteen are negative. There is also some evidence that the variability of the residuals is increasing as the number of function points increases. Four of the five largest residuals belong to the four observations with the largest function point counts: $\{1\}$, $\{2\}$, $\{19\}$, and $\{20\}$. The normal probability plot in Fig. 1(c) also suggests that normality of the residuals is suspect. In Fig. 1(d), the influence plot shows that cases $\{1\}$ and $\{2\}$ are highly influential, due in large part to their high leverage or extremeness in the independent variable. The Albrecht and Gaffney model also has the unfortunate property that function point counts of less than 245 result in the prediction of negative work-hours for completion.

The basic problem with fitting a model to this data is the relatively few cases with function point counts greater than 1200. For example, if the four cases with function point counts exceeding 1200 are removed, a very different fitted line will result. There are two possibilities to consider here: (1) the relationship between FP and E is not linear; and (2) the relationship is linear but the error variance increases with the number of function points. In the latter situation, cases with large values of FP should be down-weighted in the regression. In either event, the model as suggested by Albrecht and Gaffney is inappropriate.

The plot of residuals in Fig. 1(b) suggests that the relationship may involve a quadratic term. We might, for example, fit E versus FP^2 or $\sqrt{E}$ versus FP. The deficient normal probability plot suggests that transformation of the dependent variable, E, is the better choice.

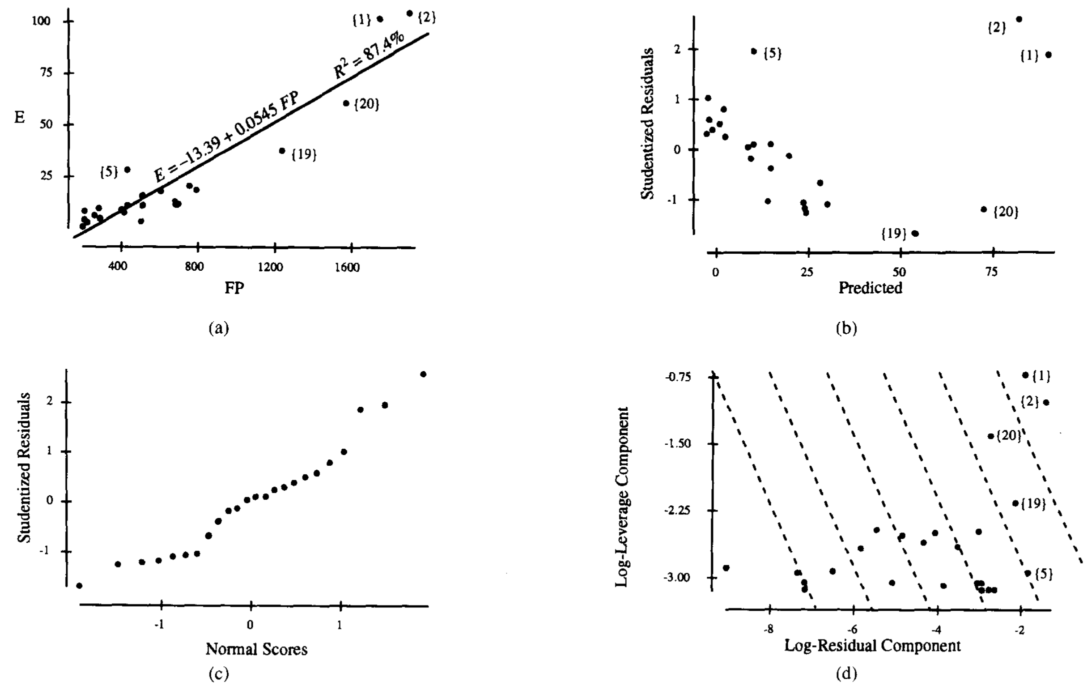


Fig. 1. Regression diagnostics for the Albrecht and Gaffney model using the IBM data. (a) Regression of Developmental Effort (E) versus Function Points (FP) Effort is measured in thousands of work-hours. (b) Studentized Residuals versus Predicted Values for the model in (a). The error terms appear nonrandom and show a strong quadratic component. (c) Normal Probability Plot of residuals for the model in (a). (d) Leverage-Residual plot of Cook's Distance. Cases that lie along the same dashed contour have the same influence. Moving from one contour line to an adjacent line represents an increase or decrease in influence by a factor to 4.

The summary analyses for this transformation are given in Fig. 2. The value of R^2 is about the same at 89.9% while the residual plot (Fig. 2(b)) supports the independence and homoscedastiscity of the error terms. The normal probability plot (Fig. 2(c)) is much improved with only case $\{5\}$ showing as unusual and in need of further investigation. The influence plot (Fig. 2(d)) shows that the influence of cases $\{1\}$ and $\{2\}$ is substantially mitigated. Of greater interest for our purposes we note that the prediction intervals for the transformed model are substantially narrower. For example, suppose we are interested in prediction of a new observation at the mean value of FP (*i.e.*, $FP = 647.625$). The original model (Albrecht and Gaffney) yields a 90% prediction interval of 3.81 to 39.93 thousands of work-hours while that of the transformed model is from 6.76 to 29.92. Of course, neither interval is especially tight due in part to the small number of observations. However, even if the Albrecht and Gaffney model produced a narrower confidence interval we would still prefer the transformed model because probability statements concerning the intervals are predicated on the model assumptions. The degree to which these are violated determines the validity of the prediction interval.

B. The Kemerer Model

Kemmerer [12] also developed a cost estimation model using function points and linear regression. The data set consists of observations from 15 software projects undertaken by a national consulting and development firm identified only as the ABC company. A scatter-plot of the data and the resulting least-squares line are shown in Fig. 3(a). The dependent variable, *Effort*, is measured in man-months where one man-month is 152 work-hours. We use this scale for ease of reference with Kemmerer [12]. Rescaling the units of the dependent variable does not impact on the analysis of the model. We also note that Kemerer reports the *adjusted* R^2 while we have used the unadjusted R^2 throughout this article. These are largely comparable measures related by the expression $1 - R_a^2 = (1 - R^2)(n - 1)/(n - p)$, where $p - 1$ is the number of predictors. The adjustment factor compensates for adding variables whose marginal contribution is small. Preference for either measure is inconsequential in practice.

It is clear from Fig. 3(a) that model fit is inappropriate in several respects. The greatest anomaly is case $\{3\}$ which is a very high leverage point and tends to pull the regression line towards itself. If case $\{3\}$ were removed, a vastly different fitted equation would result. Also, the predicted values for the smallest two projects are negative. Another shortcoming of the model is the large mean-squared-error (mse) which results in wide prediction intervals. (See the Appendix section for details.) This large mse, along with the gross violation of model assumptions renders the resulting inferences virtually meaningless.

As was the case with the Albrecht and Gaffney model, the small number of data points make it impossible to determine whether the true relationship is nonlinear, or linear but with

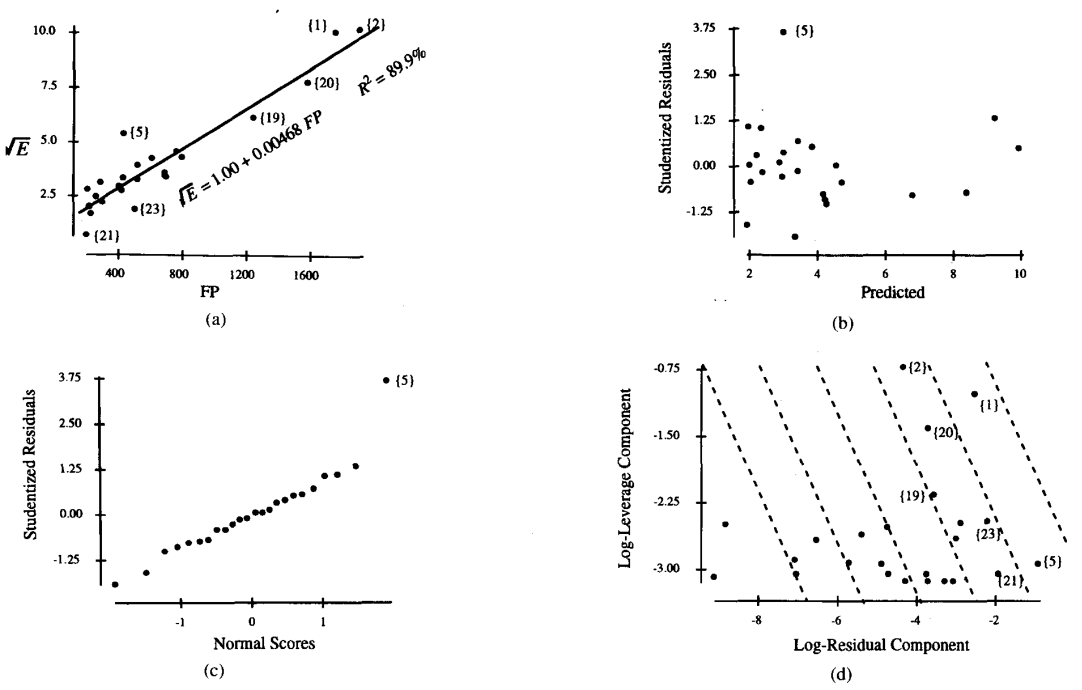


Fig. 2. Regression diagnostics for the transformed IBM data. (a) Regression of Sqrt (Developmental Effort), (E) versus Function Points (FP). Effort is measured in thousands of work-hours. (b) Studentized Residuals versus Predicted Values for the model in (a). (c) Normal Probability Plot of residuals for the model in (a). (d) Leverage-Residual plot of Cook's Distance. Cases that lie along the same dashed contour have the same influence. Moving from one contour line to an adjacent line represents an increase or decrease in influence by a factor of 4.

error variance increasing with project size. In the latter instance, case $\{3\}$ may be a rather ordinary observation. Fig. 3(b) presents two alternative polynomial models which certainly are a better fit to the data. We do not mean to suggest that either of these is the "correct" functional form of the true relationship, but rather that there are some fairly simple alternatives which provide a much better fit. Without further information about case $\{3\}$ and possibly additional data points, the relationship is most uncertain.

We also take the opportunity with this data set to point out certain common misconceptions about the importance of R^2 in model assessment. Briefly, in developing a linear model for the purpose of prediction, we are interested in a point estimate and some measure of the uncertainty of the point estimate. This is typically assessed by a confidence interval on a future value, also known as a prediction interval. (For additional details on prediction intervals as well as alternative methods of assessing predictive power, see the Appendix.) All of the usual inferential statistics are based on the assumptions of normality, independence, and homoscedasticity of the residuals. If these are seriously violated, the prediction intervals are invalid.

A regression model may be fit (that is, the least-squares estimates calculated) for *any* set of data, even if the functional form of model is entirely misspecified or there is nonconformity of the residuals to the model assumptions. Such inappropriate models may sometimes have very impressive values of R^2. As an illustration, Fig. 4 presents three regression models associated with Kemerer's ABC data. Equation (a) is

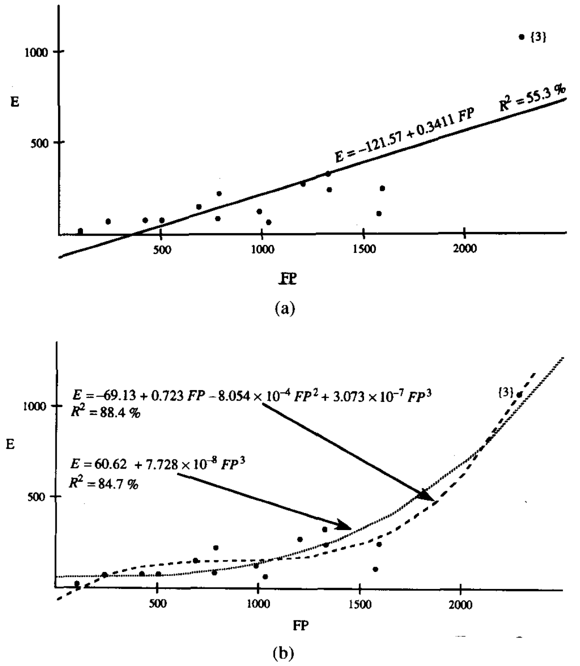


Fig. 3. Kemerer's ABC data: Effort (man-months) versus Function Points. (a) Kemerer's model for ABC data set: Effort in man-months versus Functionpoints. One man-month equals 152 hours. (b) Alternative modelsfor the ABC data: A simple cubic term model and a polynomial model of order 3.

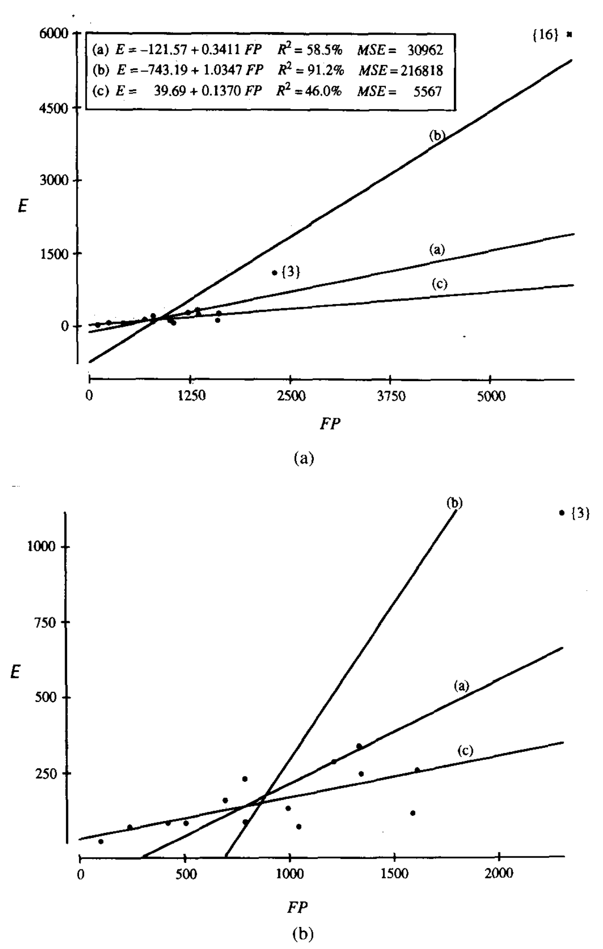


Fig. 4. Kemerer's ABC Data: Examining the Multiple Correlation Coefficient and Mean Squared Error. (a) Kemerer's ABC Data: Effort in man-months (E) versus Function Points (FP). Equation (a) is the regression line for the original 15 cases. Equation (b) is the line with case $\{3\}$ deleted and the artificial case $\{16\}$ included. Equation (c) is the line for the 14 cases exclusive of $\{3\}$ and the artificial case, $\{16\}$. (b) Enlarged view of Fig. 4(a) for the original 15 cases of the ABC data.

that reported by Kemerer for the original fifteen data points. Equation (b) is the least-squares line for the original data, excluding case $\{3\}$ but then adding an even more extreme artificial data point labeled case $\{16\}$. Alternatively, one might think of this situation as moving case $\{3\}$ to an even more extreme and outlying position. Equation (c) is the regression line for the fourteen data points exclusive of cases $\{3\}$ and $\{16\}$.

Visual inspection and intuition tell us that line (b) fits the first fourteen quite poorly in its attempt to accommodate case $\{16\}$. Yet, it has the largest R^2 among the three lines at 91.2%. In fact, by making case $\{16\}$ even more extreme, values of R^2 which are arbitrarily close to 100% may be achieved. To understand this phenomenon, recall that R^2 is the proportion of the total variability about the mean explained by the regression. As case $\{16\}$ becomes more extreme, the total variability increases, but at the same time the distance of a point from the fitted line is, relatively, a proportionally smaller amount of the distance of the point from the mean of E. Geometrically, as case $\{16\}$ becomes more extreme, the first 14 points become, by comparison, clumped together so that they behave as a single point. That is, in a relative sense, they become more and more similar. The regression is essentially fitting only "two" points: the cluster of fourteen and the extreme case $\{16\}$.

Eliminating case $\{3\}$ and case $\{16\}$ from consideration, we would anticipate the remaining 14 points would be better fit than with one of these extreme cases present. Visual inspection of line (c) supports this view. However, the value of R^2 drops to 46%. Kemerer notes that case $\{3\}$ is outlying, but upon recognizing the drop in R^2 when case $\{3\}$ is removed, he concludes that it is better to keep it in. But if we are to follow through with this reasoning, case $\{16\}$ were it present, would be even more desirable. This reasoning is completely counter intuitive and indeed, faulty.

What factors, in addition to R^2, should be considered when assessing a regression model? For the purpose of estimation, the mse plays a vital role. We observe that though it has the smallest R^2, line (c) also has the smallest associated mse (see Fig. 4). For a fixed level of confidence, the width of the prediction interval is determined primarily by mse. Hence, equation (c) is much more appealing since it will produce narrower prediction intervals over the relevant range of the independent variable. Of course, one should not summarily discard cases simply because they do not fit. The appropriate disposition or accommodation of case $\{3\}$ requires further investigation. The point is that reliance on R^2 alone for assessing the appropriateness and strength of a regression model is erroneous.

Kemerer also reports that an important finding of his study is that the Albrecht and Gaffney estimation model was validated by his independent study. Specifically, using a single regression on all $24 + 14 = 39$ data points with the units of *Effort* similarly scaled, the null hypothesis that the two models were different could not be rejected at the 95% confidence level. This finding is however, weak on two counts. First, both models are somewhat suspect, as we have demonstrated. Perhaps if case $\{3\}$ were not present in Kemerer's data, the hypothesis test would have had a different outcome. Secondly, both models are based on relatively small samples with large mse's. As a result, the respective estimated regression coefficients have large confidence intervals which make it unlikely that any differences in the models will be detected unless the differences are dramatic.

We do not mean to be overly critical of either Albrecht and Gaffney or Kemerer; each has contributed significantly to the theory and practice of software development effort measurement. We mean only to point out that the acquisition of a good regression model requires a fair amount of data as well as careful and thoughtful analysis of the underlying model assumptions.

III. Large Sample Cost Estimation Results

In this section we present the results of our analysis of function point data from 104 projects obtained from a major corporation. There were several reasons for using this

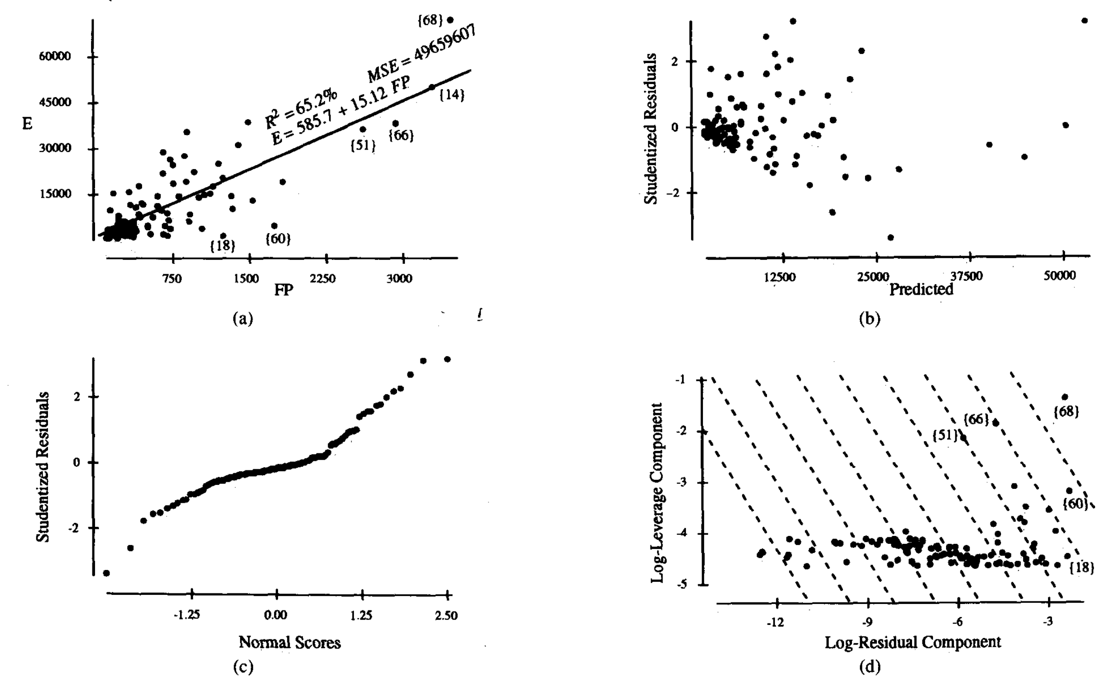


Fig. 5. Regression diagnostics for the original variables in the model (2). (a) Regression of Developmental Effort (E) versus Function Points (FP). Effort is measured in work-hours. (b) Studentized Residuals versus Predicted Values for the model in (a). (c) Normal Probability Plot of residuals for the model in (a). (d) Leverage-Residual plot of Cook's Distance. Cases that lie along the same dashed contour have the same influence. Moving from one contour line to an adjacent line represents an increase or decrease in influence by a factor of 4.

company's data. First, the company has a well-trained and experienced systems development staff and encourages the use of current software development methodologies and tools. Furthermore, the company is interested in the use of function point analysis and has used function points to measure productivity and quality for several years. The accumulated historical project data includes development effort and function points. Managers at this company were willing to provide project data and to participate in the research effort.

The set of project data represents a wide range of software applications. Included are data for systems which run on MVS and UNIX operating systems, which are programmed in COBOL, PL/I, and C, and which use IMS, IDMS, INFORMIX, INGRESS, and other data base management systems. Finally, the company has well defined procedures for reporting development hours spent on their software development projects. Since the objective of the study was the estimation of effort using a model based on actual development effort, it was important that the actual effort data be accurate and consistent. The record-keeping procedures at this company ensured that such was the case.

The 104 projects used in developing our cost estimation model represent a large sample compared to most other published studies. According to Moseley [19], "it is not uncommon to find in the literature estimation models that are based on sample sizes of fifteen to thirty projects." The projects in this study represent medium-to-large business applications, ranging in size from 119 function points to 3472 function points. Project data included the following information: software size in function points, development effort in hours, operating system, data base management system, and programming language. We will initially concentrate on using the single independent variable, FP, for prediction and subsequently discuss the inclusion of additional variables.

A scatter-plot of the data (Fig. 5(a)) suggests that a linear relationship is present and we fit our initial model,

$$E = 585.7 + 15.12\ \mathrm{FP} \tag{2}$$

where the developmental effort is given in work-hours. From the residual plot in Fig. 5(b), the variability of the error terms seems to increase as development effort increases, placing the assumption of homoscedasticity of the residuals in serious doubt. This observation also supports our intuition that the absolute precision with which we may predict the effort for large projects is less than that for small projects. For example, if we consider a large project as merely the sum of several smaller independent projects, the variance of the error terms for the sum is additive. We also note that the normal probability plot of the residuals in Fig. 5(c) departs from the straight line that one would expect were the normality assumption met.

For the data in this study, cases {68}, {14}, {66}, and {51} are high leverage points, as seen in Fig. 5(a). Fig. 5(d) is a graphical display of the leverage component, the residual component, and the resulting influence for each case. For example, case {68} has both the largest leverage component

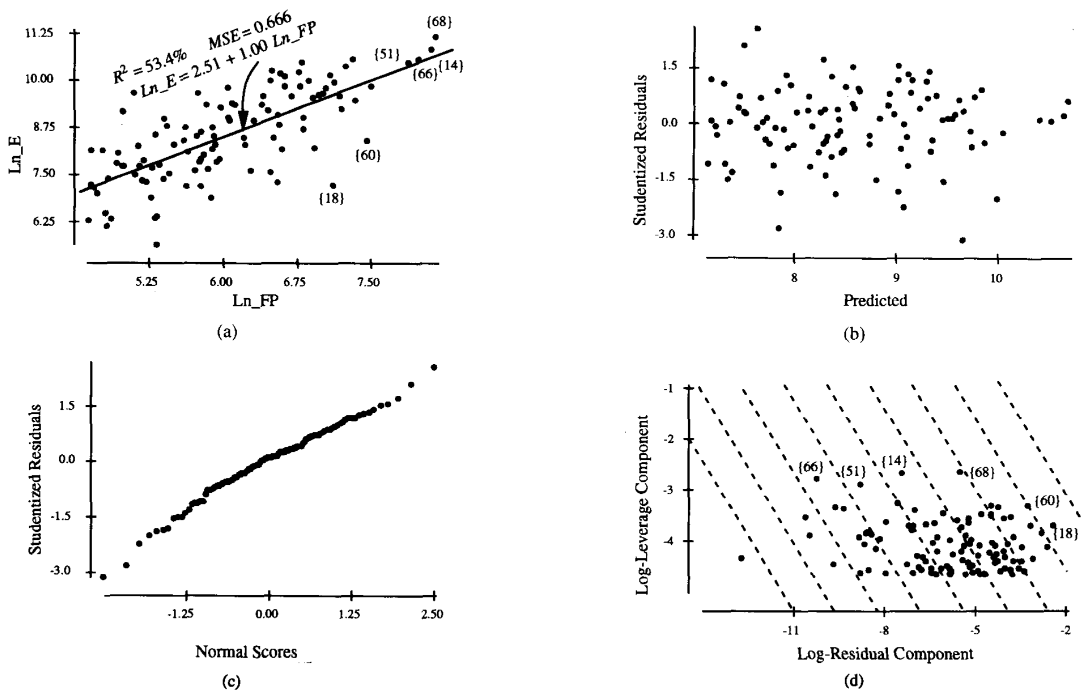


Fig. 6. Regression diagnostics for the transformed variables in the model(3). (a) Regression for the transformed variables, Ln_E versus Ln_FP. Efforts is measured in work-hours. (b) Studentized Residuals versus Predicted Values for the model in (a). (c) Normal Probability Plot of residuals for the model in (a). (d) Leverage-Residual plot of Cook's Distance. Cases that lie along the same dashed contour have the same influence. Moving from one contour line to an adjacent line represents an increase or decrease in influence by a factor of 4.

and the largest residual component, and therefore, the greatest influence. Its influence is approximately five times that of the next most influential point (case {60}) and is certainly large enough to warrant our concern. Case {14} is also a high leverage point, but its influence is negligible since its residual value is close to zero.

In order to overcome the difficulties with the model assumptions and model stability, we applied a logarithmic transformation to both the dependent and independent variables. This transformation is particularly useful in that linearity of the original relationship is preserved while stabilizing the error variance. The resulting data and fitted line,

$$\ln E = 2.51 + 1.00(\ln \text{FP}) \qquad (3)$$

are displayed in Fig. 6(a). The plots in Fig. 6(b) and 6(c) indicate that the residuals are more well behaved than those of the original model in (2). Standard tests for linearity and the normality, independence, and homoscedasticity of the residuals confirm the visual analysis.

The stability of the model is also improved. The leverage-residual influence plot in Fig. 6(d) shows that the high leverage cases are now associated with much smaller residuals, so their influence is greatly mitigated. Indeed, the most highly influential point for the transformed data is case {18}, which has an influence value one-tenth that of the most influential case of the original data (case {68}). We conclude that the model using the transformed data (3) is appropriate in all aspects and superior to the model using the original data (2).

Retrospectively, the data sets examined in Section II are now more readily understood. In view of our results here, the log-linear relationship between effort and function points may apply generally to other studies. It is only due to lack of data that the relationship remains hidden in the earlier studies. The apparent lack of linearity in each of those models was primarily a result of very few points with large FP values and an increasing error variance. For example, case {3} from Kemerer's data fits rather well into a log-log transformation. Fig. 7 shows the combined data sets of Section II (with the exception of Albrecht and Gaffney case {5}) with a log-log transformation applied. We caution that this plot is meant to illustrate only a rough, qualitative compliance with the model developed for our own data set. Without further details and reconciliation, it would be presumptuous to combine the data for any confident analysis.

Having established the aptness and stability of the model in (3), we turn our attention to its predictive power. As noted earlier in discussing the models of Albrecht and Gaffney and Kemerer, the width of the prediction interval is a more relevant measure of a model's usefulness than is R^2. For the present model (3) we first construct prediction intervals for R^2 in the usual manner and then apply the exponential function to translate the interval back to the work-hour units of the original dependent variable, E. The prediction interval arrived at in this way is not symmetric about the point estimate. For example, a new observation at FP = 2981 (i.e., ln FP = 8) yields a point estimate of 4964 work hours and a 90% prediction

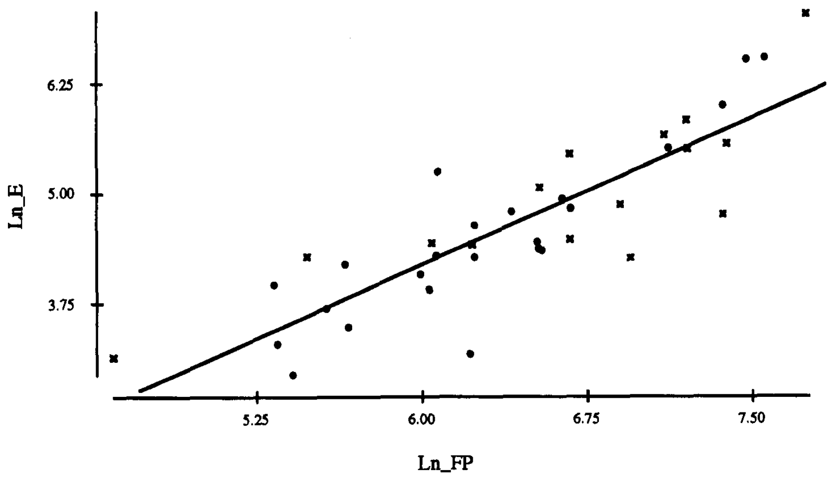


Fig. 7. Combined data sets of Albrecht and Gaffney, excluding case $\{5\}$ (denoted by •), and Kemerer (denoted by x). Regression line is that for log-log transformed data.

interval of from 1287 to 19145 work-hours. In absolute terms, larger function point values produce wider prediction intervals when translated back into the original units; but in relation to the magnitude of the point estimate, the interval widths for a given level of confidence are very nearly constant across the relevant range of function point values. Also, in presenting these results we have considered only the one-sided, upper-tail prediction value since managers are generally more concerned with possible cost overruns than coming in under budget. The one-sided intervals allow for a smaller upper bound. For example, from Table I we see that for FP = 600 the estimated work-hours is 7383 (*i.e.*, the value at $\alpha = 0.50$) with a 90% upper bound of 21275, which is 2.88 times the point estimate. A different value of FP will produce a different interval, but using this model one may say, for example, that a future project will result in work-hours less than approximately twice the estimated value with 80% probability. These prediction intervals apply to the prediction of a *single* new observation. If one is fortunate enough to have several new projects of similar size, the prediction interval for the *average* number of work-hours is significantly tighter.

TABLE I
PREDICTION INTERVALS FOR EFFORT AT $FP = 600$, FOR THE DATA SET OF SECTION III

Confidence level $(1-\alpha)$	Model (3)		Model (4)	
	Upper bound	Relative width	Upper bound	Relative width
0.95	28 822	3.90	18 984	3.44
0.90	21 275	2.88	14 419	2.61
0.85	17 355	2.35	11 989	2.17
0.80	14 770	2.00	10 359	1.87
0.75	12 866	1.74	9141	1.65
0.70	11 368	1.54	8171	1.48
0.65	10 137	1.37	7366	1.33
0.60	9094	1.23	6675	1.21
0.55	8187	1.11	6069	1.10
0.50	7383	1.00	5526	1.00

The point estimate, or value from the fitted line, for a new observation at the level $FP = 600$ using the model (3) is given by the upper bound of the 50% prediction interval. The relative widths are calculated, for example, as $21\,275/7383 = 2.88$. The value of $FP = 600$ is arbitrary. Other FP values will give different point estimates, but the relative range values are nearly constant.

We next attempted improve the fit by employing a multiple regression model which considered the additional independent variables *operating system, data base management system,* and *programming language*. With the FP variable in the model, inclusion of these additional variables resulted in only a modest improvement. Indeed, programming language and operating systems contributed almost no marginal information and only two of the four data base types were significant. The fitted multiple regression equation is given by

$$\ln E = 3.397 + 0.914(\ln \text{FP}) - 0.629\text{DB1} - 0.877\text{DB2} \quad (4)$$

where DB1 and DB2 are indicator variables for the presence or absence of the IMS and INGRESS data bases, respectively. The upper confidence bounds and relative interval lengths for this model (at the level FP = 600 and with the IMS data base present) are given in Table I. Comparison with the simple regression model in (3) shows a slight improvement.

When dealing with effort estimation models, there are other approaches to assessing the usefulness of the model for prediction which stem from the basic premise that the seriousness of prediction errors is related to the size of the project. This was, in fact the point of view we adopted in presenting our confidence interval results in Table I. That is, the width of the confidence intervals in absolute terms is larger for larger projects, but the relative widths, given in terms of the predicted effort are nearly constant for a fixed value of α.

Others have suggested similarly motivated summary measures of predictive power. One such approach, suggested by Conte, Dunsmore, and Shen [7], is to measure the (absolute) relative error. That is, the absolute difference in the observed and fitted values for Effort divided by the actual observed Effort. (See the Appendix section for further details.) These authors suggest that the average of these relative errors should be less than 0.25 for a good model. In our study, the model in (3) had an average relative error of 0.87 when transformed back to the original units while the model (4) gave a value of 0.71.

A related measure of Conte, Dunsmore, and Shen [7] is the percentage of observations whose predicted values are within a certain percentage of their actual values. In their judgement, an acceptable effort prediction model will have at least 75% of all predicted values fall within 25% of their actual observations. In our study, only 64% of the predicted values fell within 25% of their observations for the model (3) and only 68% for the model (4). Hence, by both of these relative error goodness-of-fit criteria, the function point models developed here are deemed somewhat less than adequate.

IV. Conclusion and Suggestions for Further Study

In the previous sections we have demonstrated some of the pitfalls encountered in developing good regression models and have provided a software cost estimation model based on a relatively large data set. Several questions remain. First, how useful is the model in practice within the environment in which it was developed? Secondly, how portable is the model to other organizations? And finally, can the model be improved by additional data collection?

As for the practical, same site, usefulness of the model, it seems a very broad tool at best. Many experienced managers would likely be able to estimate the final cost of a project (in hours) to within a factor of two nearly 80% of the time without the aid of function points or a cost estimation model. Though perhaps they will not estimate to within 33% of the final cost, on average, 65% of the time (see Table I). In any event, it is still preferable to be able to deliver more precision in an estimate of developmental effort.

In considering this problem, the sources of variability must be identified. Within a single organization, there are sometimes significant differences in the function point counts for the same project as determined by separate individuals. This arises from subjective assessments in both the raw counts and the adjustment factor. Suggestions for reducing or eliminating this source of variability include using a single rater for all projects or using a single *group* of raters for each project to arrive at an average [24]. Kemerer [13] and Matson and Mellichamp [17] have suggested more automated approaches to overcome this inter-rater variability. Another source of variability is the different abilities and productivity levels of various project teams. One possibility for mitigating this variability is to take personnel into account in the adjustment factor.

At a specific site or even within an organization, it may be possible to control variability as described above. However, the use of a model developed from the data of one organization for prediction at another requires more care. For example, the external organization would need to know, specifically, how work-hours were counted. Even so, it is unlikely that the external organization will be able to completely control for all inter-organizational differences that may impact the model. Therefore, we recommend that organizations using models developed elsewhere begin to collect their own data for model building. Since this may take quite some time, it may be necessary to make use of an "adopted" model from an organization which is qualitatively similar.

As we have demonstrated, function points, even in concert with other predictors, seem only moderately helpful in software development cost estimation. However, we believe that this situation may be markedly improved by making better use of available information. The function point value for a given project is a linear combination of other variables, multiplied by an adjustment factor. The method of FP calculations used for our data set takes a linear combination of 15 variables (see (1)) with predetermined coefficients which are given in Kemerer [13]. For qualitative assessment of productivity, it may be useful to combine the available information into a single measure of utility such as function points. However, for the purpose of estimation, the predetermined coefficients of the individual components unnecessarily constrain the regression to a less than optimal solution. It would be preferable to use all 15 variables (each times the adjustment factor) as predictors in the model since this will allow the estimation method to select the best weights. This will necessitate a somewhat larger minimum sample size than with just the single variable, but the potential improvement in the model may be significant. Certainly, the 104 observations in the data we have presented are sufficient for such an analysis. Unfortunately, we were unable to obtain the individual component values for this data set from which the function point counts were calculated. It may also be useful to first perform a principal components analysis to reduce the number of predictor variables from 15 or more down to a more manageable number. Even if one is intent on a single summary predictor such as FP, the first principal component will provide a set of weights for a linear combination of the components which is likely to be superior to the preassigned weights.

To illustrate the potential improvement, we return to the data of Albrecht and Gaffney, discussed in Section II. The function points calculations for this data set differ from the current definition in several respects. First, interfaces were considered as master files and not counted separately, so that there are only four basic components (inputs, outputs, master files, and inquiries) rather than five. Also, these four components were each held at only one complexity level rather than low, average, and high. Finally, the adjustment factor had a range of ±25% rather than ±35%. The resulting function point calculation for the kth observation is,

$$\mathrm{FP}_k = (4\ \mathrm{IN}\ + 5\ \mathrm{OUT}\ + 4\ \mathrm{INQ}\ + 10\ \mathrm{FILE}\ )c_k$$

where c_k is the adjustment factor and the coefficients (4, 5, 4, and 10) are those currently used for these components at the *average* level of complexity. The specific values for each of these variables are given in Table II.

In fitting a model using the four components as predictors (rather than FP) we take the position that the adjustment factor, c_k, acts multiplicatively with *each* variable. Denoting the adjusted predictors by, for example, $c_k\mathrm{IN} = \widetilde{\mathrm{IN}}$, we first

TABLE II
CONSTITUENT COMPONENTS FOR FP CALCULATIONS: ALBRECHT AND GAFFNEY DATA

Case No.	IN	OUT	INQ	FILE	ADJ	FP	EFFORT
1	25	150	75	60	1.00	1750	102.4
2	193	98	70	36	1.00	1902	105.2
3	70	27	0	12	0.80	428	11.1
4	40	60	20	12	1.15	759	21.1
5	10	69	1	9	0.90	431	28.8
6	13	19	0	23	0.75	283	10.0
7	34	14	0	5	0.80	205	8.0
8	17	17	15	5	1.10	289	4.9
9	45	64	14	16	0.95	680	12.9
10	40	60	20	15	1.15	794	19.0
11	41	27	29	5	1.10	512	10.8
12	33	17	8	5	0.75	224	2.9
13	28	41	16	11	0.85	417	7.5
14	43	40	20	35	0.85	682	12.0
15	7	12	13	8	0.95	209	4.1
16	28	38	24	9	1.05	512	15.8
17	42	57	12	5	1.10	606	18.3
18	27	20	24	6	1.10	400	8.9
19	48	66	13	50	1.15	1235	38.1
20	69	112	21	39	1.20	1572	61.2
21	25	28	4	22	1.05	500	3.6
22	61	68	0	11	1.00	694	11.8
23	15	15	6	3	1.05	199	0.5
24	12	15	0	15	0.95	260	6.1

The IBM data of Albrecht and Gaffney. The function point values are arrived at by, for example, in case {20}, $FP = [4(69) + 5(112) + 4(21) + 10(39)]1.20 = 1572$.

considered the model

$$E = \beta_0 + \beta_1 \widetilde{\mathrm{IN}} + \beta_2 \widetilde{\mathrm{OUT}} + \beta_3 \widetilde{\mathrm{INQ}} + \beta_4 \widetilde{\mathrm{FILE}} + \varepsilon.$$

However, this model resulted in a fit which exhibited many of the same shortcomings as the original Albrecht and Gaffney model. After some exploration, we selected the model in which each predictor variable is transformed by taking its square. The fitted equation for the transformed data is

$$E = 3.80 + 0.00119\, \widetilde{\mathrm{IN}}^2 + 0.00210\, \widetilde{\mathrm{OUT}}^2 + 0.00608\, \widetilde{\mathrm{INQ}}^2 + 0.0045\, \widetilde{\mathrm{FILE}}^2 \qquad (5)$$

This model is superior in several respects. The residuals are very well behaved, although, as before, case {5} appears to be an outlier in need of further investigation. The value of R^2 is 97.5% and the value of mse is 25.7 which is less than one-fourth that of the Albrecht and Gaffney model. This results in prediction intervals less than one-half as wide.

We suspect that the functional form for the model in (5) may turn out differently if more data were available and we cannot recommend any model, based on such a small sample size. The key point is that improved results may be achieved by unbundling the function point variable into its constituent components. This direction seems to offer a strong possibility of more precise prediction in future studies.

APPENDIX
ASSESSING THE PREDICTIVE POWER OF A REGRESSION MODEL

One of the primary uses of any regression model is in the prediction of future values of the dependent variable at some specified levels of the independent variables. For ease of discussion, consider the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, i = 1, \cdots, n \qquad \text{(A1)}$$

where the error terms ε_i are assumed to be independent, normally distributed random variables with zero mean and identical variance, σ^2. (Generalizations to more than one independent variable are straightforward.) The least-squares fitted model is denoted by $\hat{Y}_i = b_0 + b_1 X_i$, with residuals $e_i = (Y_i - \hat{Y}_i)$.

Now, the true line in (A1) is unknown and $\hat{Y}_h$ is a point estimate of the true mean at level h. Also, the estimated variance of $\hat{Y}_h$ is given by

$$s^2_{\hat{Y}_h} = \mathrm{mse}\left(\frac{1}{n} + \frac{(X_h - \overline{X})^2}{\sum_{i=1}^n (X_i - \overline{X})^2}\right) \qquad \text{(A2)}$$

where mse $= \frac{1}{n-2}\sum_{i=1}^n e_i^2$ is the sample estimate of σ^2.$\overline{\mathrm{MRE}}(1 - \alpha)100\%$ confidence interval on the true mean value at level h is $\hat{Y}_h \pm t_{\alpha/2,n-2} s^2_{\hat{Y}}$.

When the goal is the prediction of some single future value at level X_h rather than the mean, one must consider two sources of variability. First is the vaibility associated with the location of the true mean, which is given in (A2) and secondly, the variability for the probability distribution of a single value about its mean, which is σ^2, estimated by mse. Hence, the estimated variance for a future value at level h

is mse + $S^2_{\hat{Y}}$ and the corresponding $(1-\alpha)$ 100% prediction interval is $\hat{Y}_h \pm t_{\alpha/2,n-2}(\text{mse} + s^2_{\hat{y}})$. From the expression in (A2) we note that values X_h far away from $\overline{X}$ result in larger variances and hence wider prediction intervals.

The width of the prediction intervals gives an indication of the usefulness of the model in assessing future values. For a fixed level of α, a sample size, n, and a given level of future prediction, h, this width is controlled by the value of mse. As illustrated in Section III, mse is sometimes more indicitive of model usefulness than is R^2.

There are other approaches to assessing the acceptability of effort prediction models. Conte, Dunsmore, and Shen [7] introduced a measure, the mean magnitude of relative error, defined by

$$\overline{\text{MRE}} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{Y_i}\right| = \frac{1}{n}\sum_{i=1}^{n}\text{MRE}. \quad \text{(A3)}$$

The implicit assumption in this summary measure is that the seriousness of the absolute error is proportional to the size of the observation. For effort estimation models, this seems reasonable. For example, an ablolute error of four days on a small project may be comparable to an absolute error of several weeks on a much larger project. In this setting, the use of relative error may be more appealing. These same authors report that they consider a value of $\overline{\text{MRE}} \leq 0.25$ to be acceptable for estimation effort models.

A companion summary measure related to $\overline{\text{MRE}}$ is the prediction at level ℓ, $\text{PRED}(\ell) = \frac{k}{n}$, where k is the number of observations whose MRE is less than or equal to ℓ and n is the sample size. Conte, Dunsmore, and Shen [7] conclude that a good effort estimation model should have $\text{PRED}(0.25) \geq 0.75$. That is, 75% of the fitted or predicted values should fall within 25% of their actual observations.

These methods are *ad hoc* procedures which make no assumption concerning the distribution of the observations. They are specifically suited to the situation where the errors are increasing with the size of the observation. Indeed, if the model assumptions in (A1) have been met, the relative error measures are inappropriate since the error variance is constant. In dealing with the problem of increasing variance, (i.e., residuals), one approach is to transform the variables in the model in order to meet the assumptions in (A1). This was the approach taken in this article. In this instance, the calculation of the relative error measures $\overline{\text{MRE}}$ and $\text{PRED}(\ell)$ should be done using the original observations and the corresponding fitted values which have been transformed back into the original units.

References

[1] A.J. Albrecht, "Measuring application development productivity," in *Programming Productivity: Issues for the Eighties*, C. Jones, ed. Washington, DC: IEEE Computer Society Press, 1981.

[2] A.J. Albrecht and J.E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Software Eng.*, vol. SE-9, no. 6, pp. 639–648, June 1983.

[3] B.E. Barrett and R.F. Ling, "General classes of influence measures for multivariate regression," *J. American Statistical Assoc.*, vol. 87, pp. 184–191, 1992.

[4] B.W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[5] B.W. Boehm and P.N. Papaccio, "Understanding and controlling software costs," *IEEE Trans. Software Eng.*, vol. 14, pp. 1462–1477, Oct. 1988.

[6] R.D. Cook, "Detection of influential observations in linear regression," *Technometrics*, vol. 19, pp. 15–18, 1977.

[7] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, *Software Engineering Metrics and Models*. Menlo Park, CA: Benjamin Cummings, 1986.

[8] R.D. Emrick, "In search of a better metric for measuring productivity of application development," *Int. Function Point Users Group Conf. Proc.*, 1987.

[9] D. Ince, "Software metrics," *Measurement for Software Control and Assurance*, B. A. Kitchenham and B. Littlewood, eds. New York: Elsevier, 1989.

[10] C. Jones, *Programming Productivity*. New York: McGraw-Hill, 1986.

[11] ——, "A short history of function and feature points," *Int. Function Point Users Group Conf. Proc.*, 1988.

[12] C.F. Kemerer, "An empirical validation of software cost estimation models," *Commun. ACM*, vol. 30, no. 5, pp. 416–429, 1987.

[13] ——, "Reliability of function points measurements," *Commun. ACM*, vol. 36, pp. 85–97, 1993.

[14] L.A. Laranjeira, "Software size estimation of object-oriented systems," *IEEE Trans. Software Eng.*, vol. 16, pp. 64–71, Jan. 1990.

[15] W.E. Lehder, D.P. Smith, and W.D. Yu, "Software estimation technology," *AT&T Tech. J.*, vol. 67, pp. 10–18, July–Aug. 1988.

[16] G.C. Low and D.R. Jeffery, "Function points in the estimation and evaluation of the software process," *IEEE Trans. Software Eng.*, vol. 16, pp. 64–71, 1990.

[17] J.E. Matson and J.M. Mellichamp, "An object-oriented tool for function point analysis," *Expert Syst.*, vol. 10, pp. 3–14, Feb. 1993.

[18] C.E. McCulloch and D. Meeter, "Discussion of Outliers," by R. J. Beckman and R. D. Cook, *Technometrics*, vol. 25, pp. 152–155, 1983.

[19] C.W. Moseley, "A timescale estimating model for rule-based systems," Ph.D. diss., North Tex. State Univ., 1987.

[20] B. Porter, "A critical comparison of function point counting techniques," *Int. Function Point Users Group Conf. Proc.*, 1988.

[21] C. Richards, "Estimating function points," *Int. Function Point Users Group Conf. Proc.*, 1989.

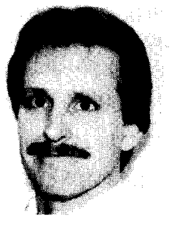
[22] H.A. Rubin, "Macro-estimation of software development parameters: The ESTIMACS system," *SOFTFAIR Conf. Software Dev. Tools, Techniques, and Alternatives*, 1983, pp. 109–118.

[23] J. Sprouls, IFPUG Function Point Counting Practices Manual Release 3.0, Int. Function Point Users Group, Westerfield, OH, 1990.

[24] C.R. Symons, "Function point analysis: Difficulties and improvements," *IEEE Trans. Software Eng.*, vol. 14, pp. 2–11, Jan. 1988.

J.E. Matson received the M.S. in industrial engineering from Mississippi State University and the Ph.D. degree in management science from the University of Alabama.

He is an Assistant Professor in the Department of Industrial Engineering at the University of Alabama. In addition, he has more than 10 years of industrial experience with AT&T and Southern Bell. His research interests include computer systems engineering and simulation.



B.E. Barrett received the B.S. in biology from the College of Charleston and the M.S. and Ph.D. degrees in mathematical science from Clemson University, Clemson, SC.

He is Assistant Professor of statistics at the Manderson School of Business, University of Alabama, Tuscaloosa, AL. His primary areas of research interest are regression diagnostics and statistical computing.

Dr. Barrett has published articles in *Journal of the American Statistical Association*, *Journal of Computational and Graphical Statistics*, and *Statistics and Computing*.

J. M. Mellichamp received the B.S. degree in industrial engineering from the Georgia Institute of Technology and the Ph.D. degree in engineering management from Clemson University, Clemson, SC.

He is a Professor of management science at the Manderson Graduate School of Business, University of Alabama, Tuscaloosa, AL. His primary areas of research are simulation and knowledge-based systems, with an emphasis in the use of knowledge-based systems and simulation methods to design complex systems.

Dr. Mellichamp has published numerous articles in such journals as *Management Science, Simulation, Expert Systems, Interfaces, Decision Sciences, International Journal of Production Research, Harvard Business Review,* and IEEE NETWORKS. He is a member of the Institute of Management Science and the Society for Computer Simulation.